Table of Contents

PBS on Columbia **1**
Overview 1
Resources Request Examples 2
Default Variables Set by PBS 4
Sample PBS Script for Columbia 5

PBS on Columbia

Overview

DRAFT

This article is being reviewed for completeness and technical accuracy.

On Columbia, PBS (version 9.2) is used to manage batch jobs that run on the four compute systems (Columbia21-24). PBS features that are common to all NAS systems are described in other articles. Read the following articles for Columbia-specific PBS information:

- queue structure

- resource request examples
- default variables set by PBS
- sample PBS scripts

Resources Request Examples

DRAFT

This article is being reviewed for completeness and technical accuracy.

All of the Columbia compute engines, Columbia21-24, are single system image Altix 4700 systems:

```
Columbia21 (508 CPUs total, 1.8 GB memory/CPU through PBS)
Columbia22 (2044 CPUs total, 1.8 GB memory/CPU through PBS)
Columbia23 (1020 CPUs total, 1.8 GB memory/CPU through PBS)
Columbia23 (1020 CPUs total, 1.8 GB memory/CPU through PBS)
```

Here are a few examples of requesting resources on Columbia:

Example 1:

If your job needs fewer than 508 CPUs and you do not care which Columbia system to run your job on, simply use *ncpus* to specify the number of CPUs that you want for your job. For example:

```
#PBS -l ncpus=256
```

Example 2:

If you specify both the *ncpus* and *mem* for your job, PBS will make sure that your job is allocated enough resources to satisfy both *ncpus* and *mem*. For example, if you request 4 CPUs and 14 GB of memory, your job will be allocated 8 CPUs and 14.4 GB because the amount of memory associated with 4 CPUs is not enough to satisfy your memory request.

```
#PBS -l ncpus=4,mem=14GB
```

Example 3:

If you want your job to run on a specific Columbia machine, for example, Columbia22 with 256 CPUs, use

```
#PBS -l select=host=columbia22:ncpus=256
```

Note that the ncpus request must appear with the select=host request and must not be present as a separate request either on the qsub command line or in the PBS script.

Example 4:

If you ever need to run a job across two Columbia systems, for example, 508 CPUs on one Columbia and another 508 CPUs on another, use

```
#PBS -l select=2:ncpus=508,place=scatter
```

Default Variables Set by PBS

DRAFT

This article is being reviewed for completeness and technical accuracy.

You can use the "env" command--either in a PBS script or from the command line of an interactive PBS session--to find out what environment variables are set within a PBS job. In addition to the PBS_xxxx variables, the following ones are useful to know.

- NCPUS defaults to number of CPUs that you requested.
- OMP_NUM_THREADS defaults to 1 unless you explicitly set it to a different number.

  If your PBS job runs an OpenMP or MPI/OpenMP application, this variable sets the number of threads in the parallel region.
- OMP_DYNAMIC defaults to *false*.

  If your PBS job runs an OpenMP application, this disables dynamic adjustment of the number of threads available for execution of parallel regions.
- MPI_DSM_DISTRIBUTE defaults to *true*.

  If your PBS job runs an MPI application, this ensures that each MPI process gets a unique CPU and physical memory on the node with which that CPU ist is associated.
- FORT_BUFFERED defaults to 1.

  Setting this variable to 1 enables records to be accumulated in the buffer and flushed to disk later.

Sample PBS Script for Columbia

DRAFT

This article is being reviewed for completeness and technical accuracy.

```
#PBS -S /bin/csh
#PBS -N cfd
#PBS -l ncpus=4
#PBS -l mem=7776MB
#PBS -l walltime=4:00:00
#PBS -j oe
#PBS -W group_list=g12345
#PBS -m e

# By default, PBS executes your job from your home directory.
# However, you can use the environment variable
# PBS_O_WORKDIR to change to the directory where
# you submitted your job.

cd $PBS_O_WORKDIR

# For MPI jobs, there is an SGI MPT module loaded by default, unless you
# modify your shell start up script to unload it or switch to a different
# version. You can use either mpiexec or mpirun to start your job.

mpiexec -np 4 ./a.out < input > output

# It is a good practice to write stderr and stdout to a file (ex: output)
# Otherwise, they will be written to the PBS stderr and stdout in /PBS/spool
# which has limited amount  of space. When /PBS/spool is filled up, any job
# that tries to write to /PBS/spool will die.

# -end of script-
```